Data science and Economics, Department of Economics, Management, and Quantitative Methods

UNIVERSITÀ DEGLI STUDI DI MILANO

IMAGE CLASSIFICATION WITH ARTIFICIAL NEURAL NETWORK

Glasses or no Glasses

Name: *Natasha Amjad*

Mat° *939736*

“I/We declare that this material, which I/We now submit for assessment, is entirely my/our work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offenses in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study. “

Table of Contents

Table of Figures

1. Summary:

This project aims to build a classifier for detecting faces wearing glasses vs no glasses. The dataset used in this project is retrieved from Kaggle. The dataset is generated using Generative Adversarial Neural Network (GAN), which consists of feature vectors of images of individuals wearing glasses or not wearing glasses. The GAN network creates these images using 512 number latent vectors.

In this work we trained Artificial Neural Network for binary classification and take a look at subsequent questions:

- How the accuracy of the Neural Network changes with the complexity of the model?
- How well the model performs with the tuning of various hyperparameters?

To answer these questions, we build three different Neural Network models. For the aim of this project, we begin with a small network and gradually increase the number of layers evaluating change in performance and the potential overfitting as the network becomes larger and more complex. To prevent problems like underfitting and overfitting, a series of techniques such as Dropout and regularization is used.

2. Dataset and Pre-processing:

The GAN dataset is taken from Kaggle online community. The GAN CSV file is used as a training dataset. The training dataset consists of 4500 rows where each row represents a training image with 514 columns. In the last column, we have the feature we would like to predict. We stored the dataset in a data frame and convert It into an array. The dataset has been split into train and test datasets with a size of 30% and 70%. It is free from NULL and duplicate values. The data type for latent vectors is a float which suits our requirement so there is no need for pre-processing and data is well built. The distribution of the target class is shown below. exclusive

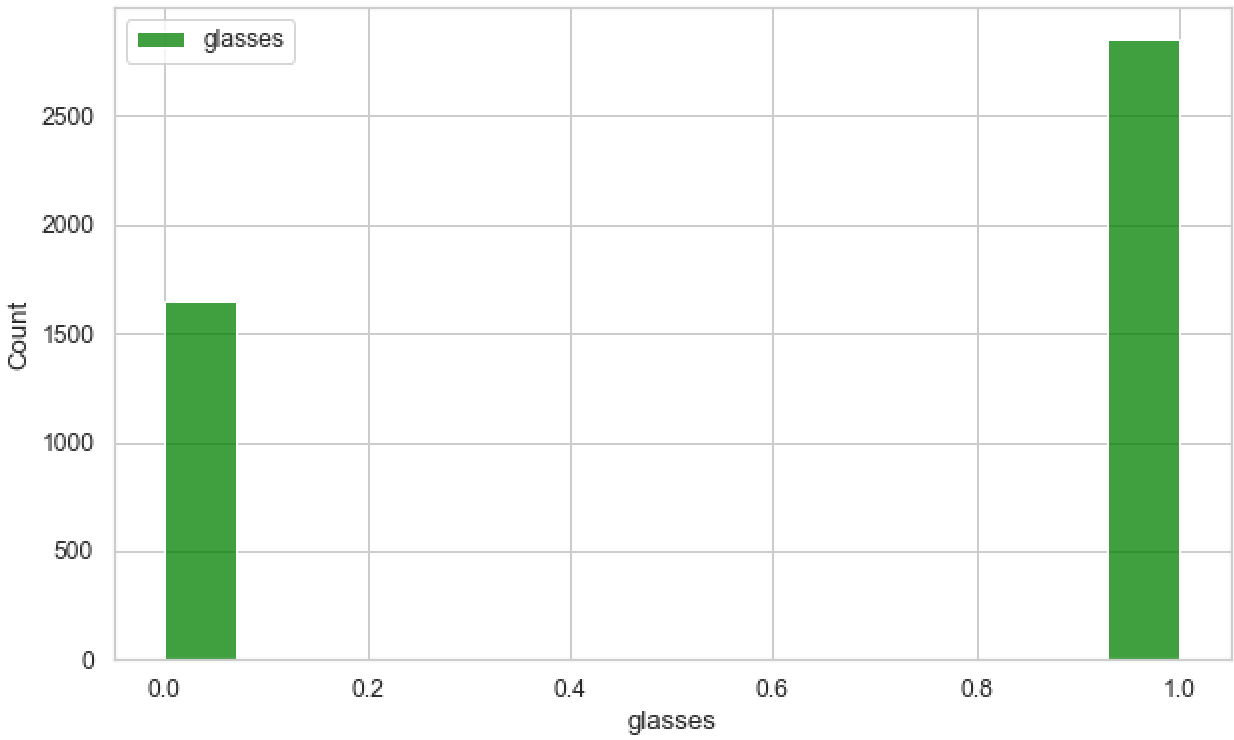


Figure 2-1:*The distribution of target class*

The dataset split is consisted of:

- X train: (3150, 512)
- X test: (1350, 512)
- Y train: (3150,)
- Y test: (1350,)

3. Activation and loss Function:

In the architecture implemented, the algorithm is trained using binary cross-entropy. The optimizer used to train the network is SGD in the first model and Adam for the rest of them. The activation function used in hidden layers is ReLU. ReLU is chosen because, at an empirical level, it is one of the most popular among researchers. When we are trying to predict the output of binary classification, we use the sigmoid activation function at the output layer. It squeezes all the initial outputs to be between 0 and 1.

4. Artificial Neural Network Model 1:

In this model we created Neural Network architecture using three layers:

- One input layer
- Two hidden layers with 32 neurons
- One output layer with 1 neuron

The hidden layers use the ReLU activation function. It is half rectified, that is for all input less than zero or negative the value is 0, and for anything positive, the value is retained. In the output layer, we used the sigmoid function which holds the values between 0 and 1 into the form of a sigmoid curve.

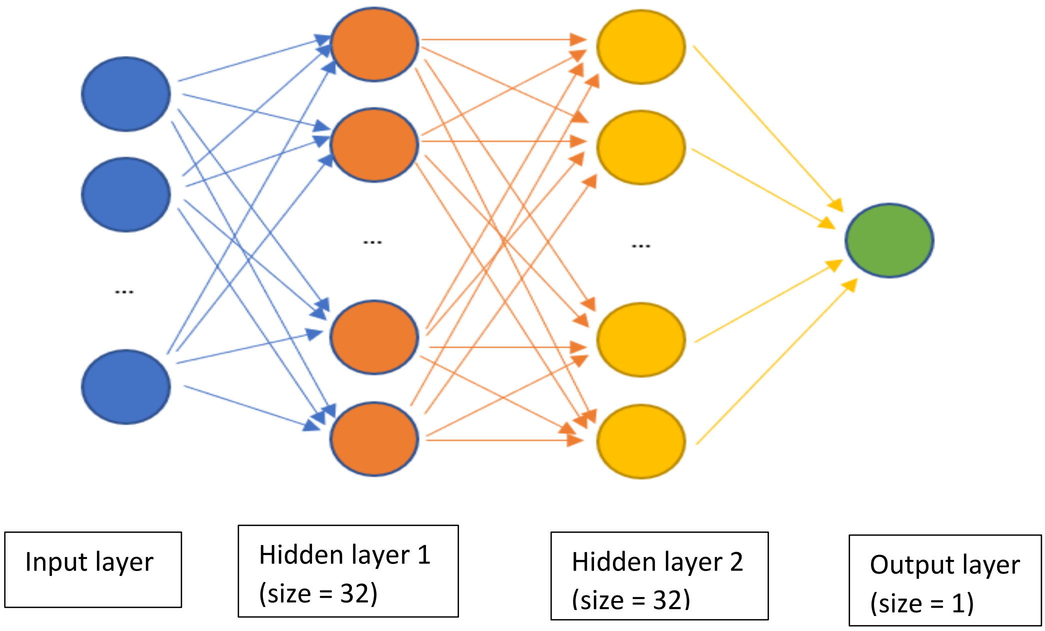


Figure 4-1: Neural Nets Architecture with two hidden layers

After being defined the Neural Nets architecture we compile the model by using the ‘side optimizer. The model is trained with 10 epochs and batch-size 32. The model accuracy and loss for validation and training set are shown below. As we can see the loss is decreasing and accuracy increasing over time. In this model, we can see the underfitting problem as the training loss continues to decrease until the end of training.

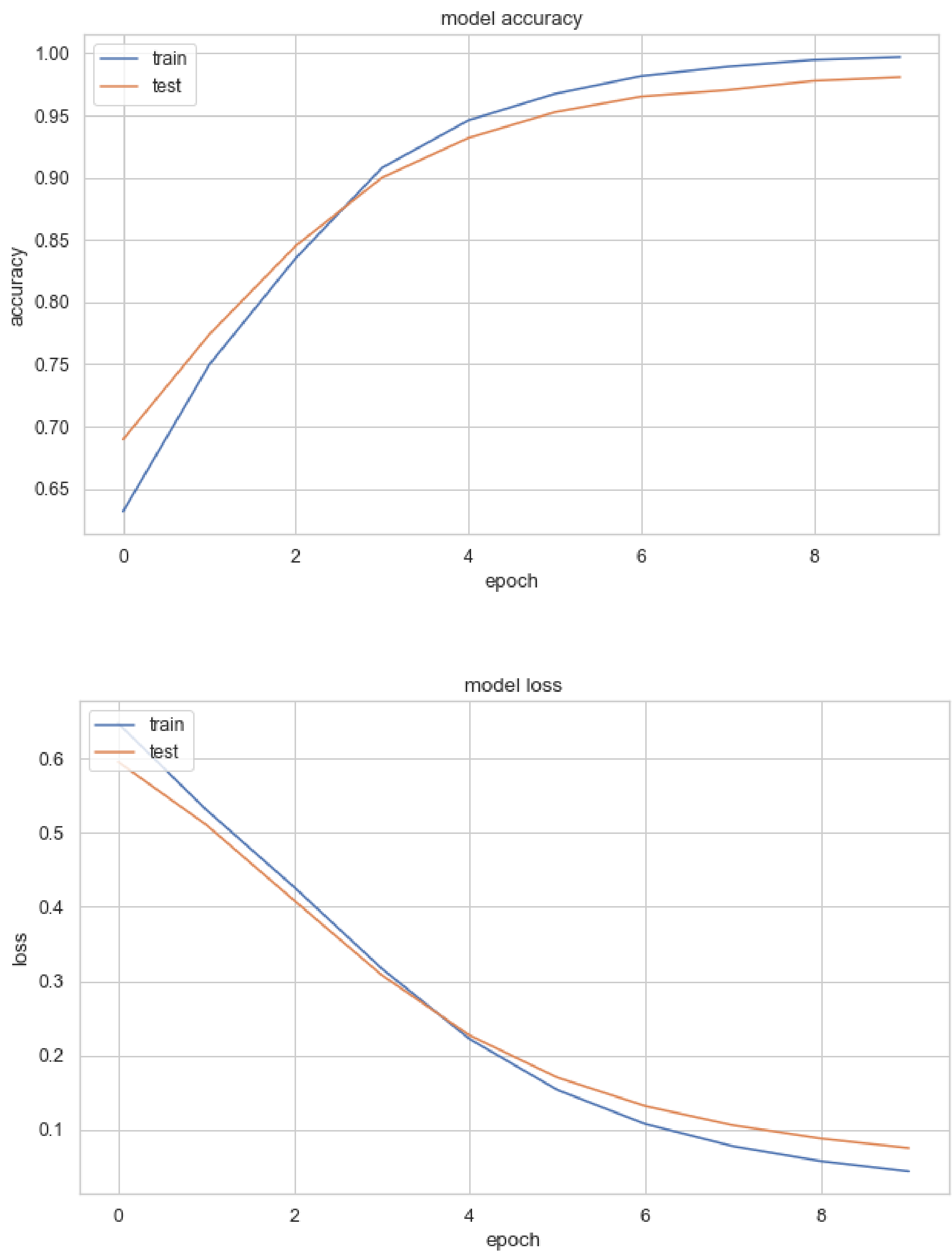


Figure 4-2: Accuracy and loss for training and validation set

Performance evaluation:

To evaluate the performance of our classifier we built a confusion matrix. As we can see our model is performing well overall but still has a little high rate of False-positive and False-negative values. We computed the F1 score (98%), which is harmonic to precision and recall. It takes both FN and FP values into account.

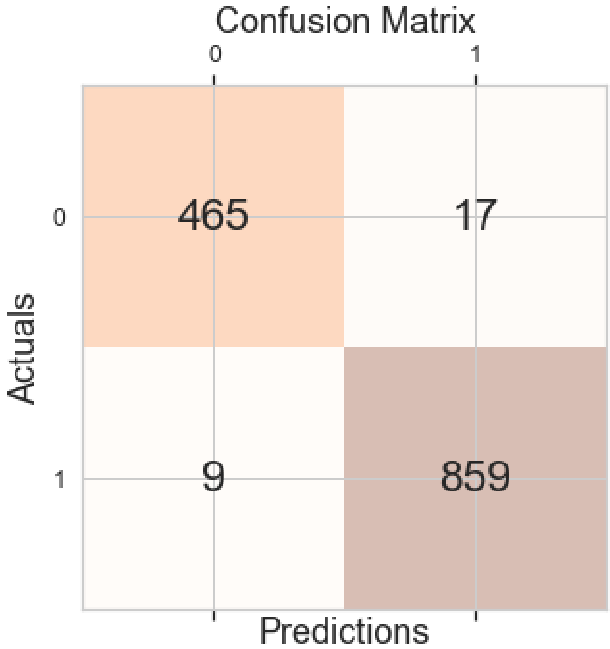


Figure 4-3: confusion matrix of NN model 1

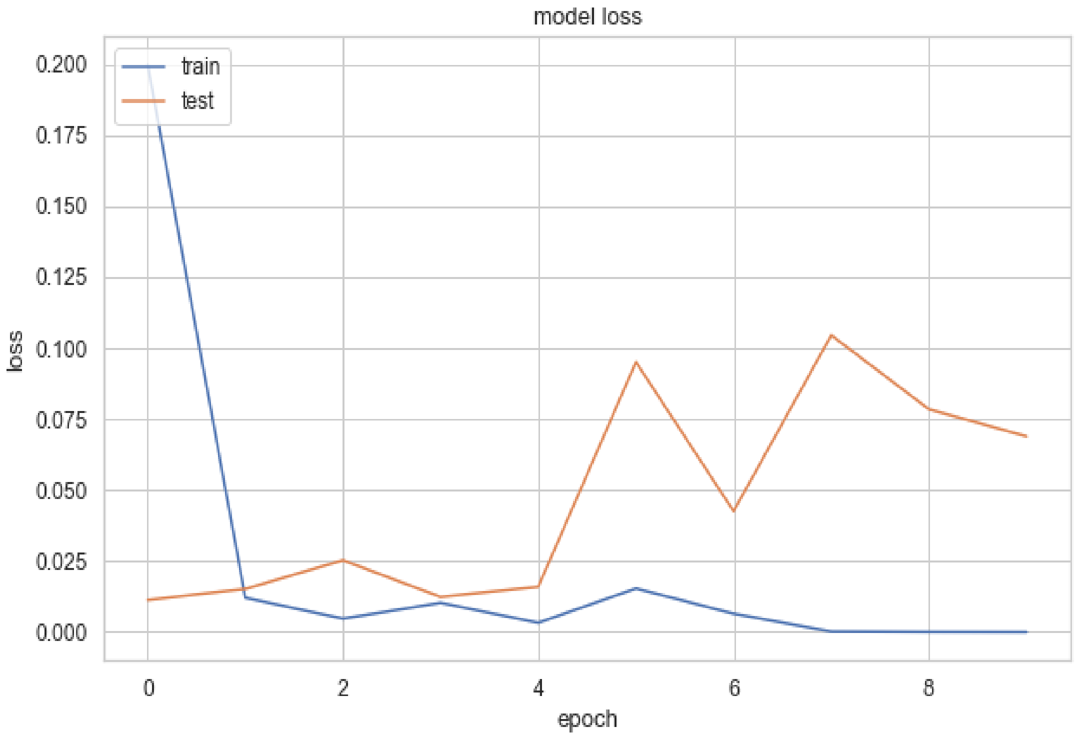
5. Neural Network Model 2:

As compared to the first model, here we built more dense Neural Nets architecture. We used 'Adam' which is one of the most common optimizers. It adds some twists to stochastic gradients descendent as it reaches the lower loss function faster.

Our second model consists of:

- One input layer
- Four hidden layers using ReLU function
- One output layer

To check how the accuracy changes with a denser network we built our architecture with more layers than the previous one. Each hidden layer makes up 1000 neurons. We trained our model using the same batch size and epochs. As we can see in the graph given below the validation loss is increasing after a certain point which is an overfitting problem.



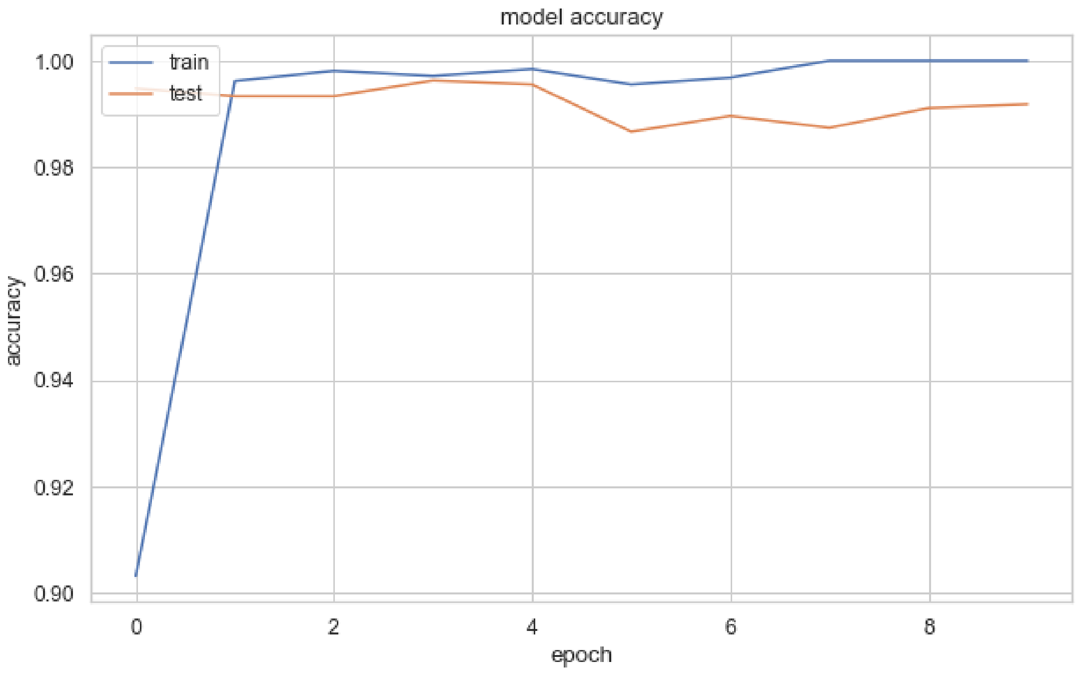


Figure 5-1: Accuracy of training and validation set in NN model 2

After training our second model we attained higher accuracy (99%) than before. As expected, the training accuracy is always higher than the validation accuracy while the opposite occurs for loss.

Performance evaluation:

As the accuracy improved with the training denser model, we also have achieved better model performance.

- Precision = 0.98
- Recall = 1
- F1 score = 0.99

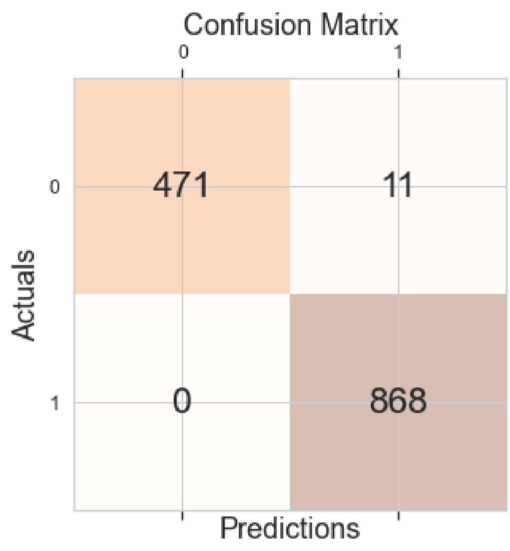


Figure 5-2: Confusion Matrix of NN Model 2

6. Neural Network Model 3:

Following the previous model, in which we have seen the overfitting problem we built a third model with some strategies to minimize the problem. We incorporated L2 regularization and dropout in each layer. Early stopping also uses to reduce the overfitting problem but we didn’t use it here because our validation loss didn’t take U shape, so it will not be effective to use it to overcome this problem. We set the L2 value to 0.01, this adds the squared values of the parameter in our loss function, and weight them by 0.01 in the loss function.

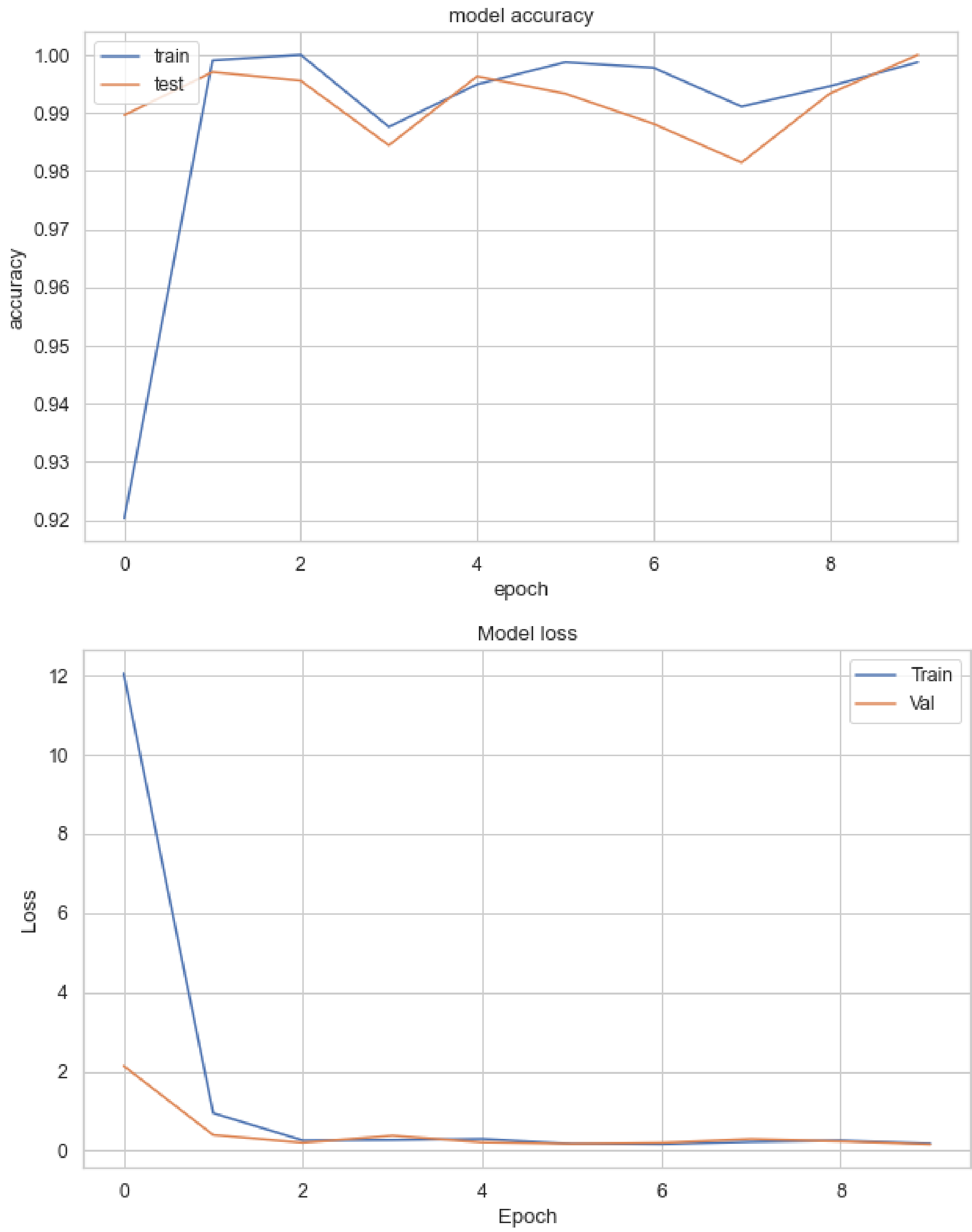


Figure 6-1: Accuracy and loss for training and validation set in NN model 3

We added a dropout of 0.3 in each layer this means that the neuron in the previous layer has a probability of 0.3 in dropping out in the training set. We compile this model with the same hyperparameters as in the previous model. As we can see in Figure 6.1 the loss at the start is quite higher because we use changed our loss function but applying these strategies, we minimized the overfitting problem. In this model, we attained accuracy (100%).

Performance evaluation:

The performance of this model is quite incredible. By applying different strategies, we improved not only the accuracy but also the performance of our model.

- Precision = 1.00
- Recall = 1.00
- F1 score = 1.00

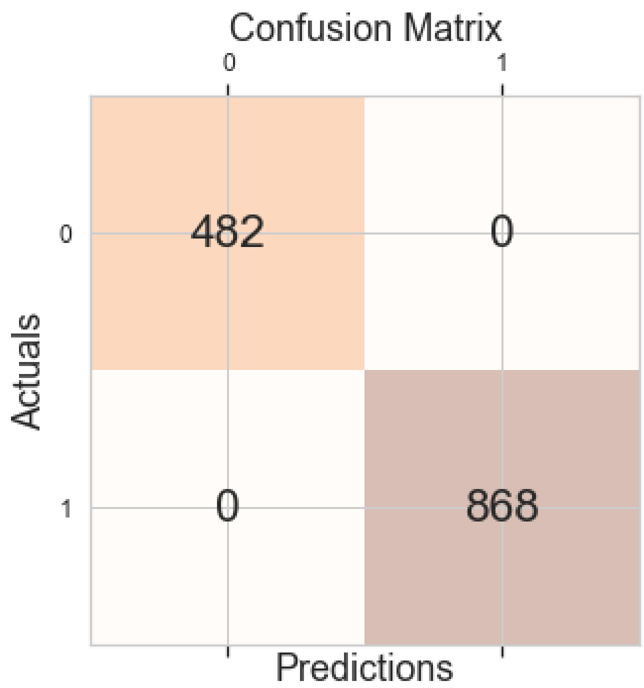


Figure 6-2:Confusion Matrix of NN Model 3

7. Conclusion:

In the light of findings and observations of our research, we can conclude that:

- The model that returns the best solution is 3rd one, which uses regularization and dropout layers with the dense network.
- Starting the experiment with simpler Neural Network architecture and expanding their complexity is a good strategy to find the best fit among various architectures.
- We noticed that higher complexity in the model improved the performance, but it is necessary to calibrate the complexity of the network to avoid overfitting/underfitting problems.
- We assumed that the hyperparameters play a vital role in training good Neural networks, such as Adam performed better than the SGD optimizer.

8. Bibliography:

https://www.researchgate.net/publication/317286513_A_Feed-Forward_Backpropagation_Neural_Network_Method_for_Remaining_Useful_Life_Prediction_of_Francis_Turbines

https://medium.com/intuitive-deep-learning/intuitive-deep-learning-part-1a-introduction-to-neural-networks-d7b16ebf6b99

https://tutorials.one/how-to-use-learning-curves-to-diagnose-machine-learning-model-performance/

Freecodecamp.org

towardsdatascience.com